

A Crash Course in Automatic Differentiation

Uwe Naumann

Mathematics and Computer Science Division

Argonne National Laboratory

- Introduction
- Forward and Reverse Modes
- Tool Development and Handout
- Advanced Issues
- Summary

Automatic Differentiation – Introduction

- Technique for efficient computation of derivatives for numerical simulation programs with machine accuracy (**no divided differences**)
- Based on local partial derivatives and chain rule
- ≈ 200 people world-wide; 3 workshops [5,6,7]; “the book” by Griewank [9]; ≥ 10 software tools
- Internet portal

<http://www.autodiff.org/>

Automatic Differentiation – Some Maths

Given a computer program for

$$F : \mathbb{R}^{n+\tilde{n}} \rightarrow \mathbb{R}^{m+\tilde{m}}, \quad (\mathbf{x}, \tilde{\mathbf{x}}) \mapsto (\mathbf{y}, \tilde{\mathbf{y}}) = F(\mathbf{x}, \tilde{\mathbf{x}})$$

for example, $\boxed{y = a * \sin(x) - x * x}$, we are interested in

$$\dot{Y} = F'(\mathbf{x}, \tilde{\mathbf{x}}) \cdot \dot{X} \quad \text{and/or} \quad \bar{X} = \bar{Y} \cdot F'(\mathbf{x}, \tilde{\mathbf{x}}),$$

where $\dot{X} \in \mathbb{R}^{n \times q_1}$, $\dot{Y} \in \mathbb{R}^{m \times q_1}$ and $\bar{Y} \in \mathbb{R}^{q_2 \times m}$, $\bar{X} \in \mathbb{R}^{q_2 \times n}$.

The Jacobian matrix is defined as

$$F' = \left(\frac{\partial y_j}{\partial x_i} \right)_{i=1, \dots, n}^{j=1, \dots, m}$$

Automatic Differentiation – Some Insight

$$\dot{Y} = F'(\mathbf{x}, \tilde{\mathbf{x}}) \cdot \dot{X}$$

$$\bar{X} = \bar{Y} \cdot F'(\mathbf{x}, \tilde{\mathbf{x}})$$

\mathbf{n}

\mathbf{n}

\mathbf{m}

$$\mathbf{m} = \begin{bmatrix} & \\ & \textcolor{red}{1} \\ & \end{bmatrix} \cdot \begin{bmatrix} & \\ & \textcolor{red}{0} \\ & \end{bmatrix}$$

\mathbf{q}

$$\mathbf{q} = \begin{bmatrix} & \\ & \textcolor{red}{1} \\ & \end{bmatrix} \cdot \begin{bmatrix} & \\ & \textcolor{red}{0} \\ & \end{bmatrix}$$

\mathbf{n}

$$\begin{bmatrix} & \\ & \textcolor{red}{1} \\ & \end{bmatrix} \cdot \begin{bmatrix} & \\ & \textcolor{red}{0} \\ & \end{bmatrix}$$

$$\dot{Y}(j, i) = F'(:, j) \cdot \dot{X}(:, i)$$

$$\bar{X}(j, i) = \bar{Y}(j, :) \cdot F'(:, i)$$

$$F' = F' \cdot I_n = I_m \cdot F'$$

⇒ cheap gradients!

Linearized Single Assignment Form (of $y=a\sin(x)-x*x$)

$$v_j = \varphi_j(v_i)_{i \prec j}, \quad j = 1, \dots, n + p + m,$$

for example,

$$\boxed{v1=x; \quad v2=\sin(v1); \quad v3=a*v2; \quad v4=v1*v1; \quad v5=v3-v4; \quad y=v5}.$$

We assume that the elemental functions are differentiable in some neighborhood of the current argument and we denote the local partial derivatives with

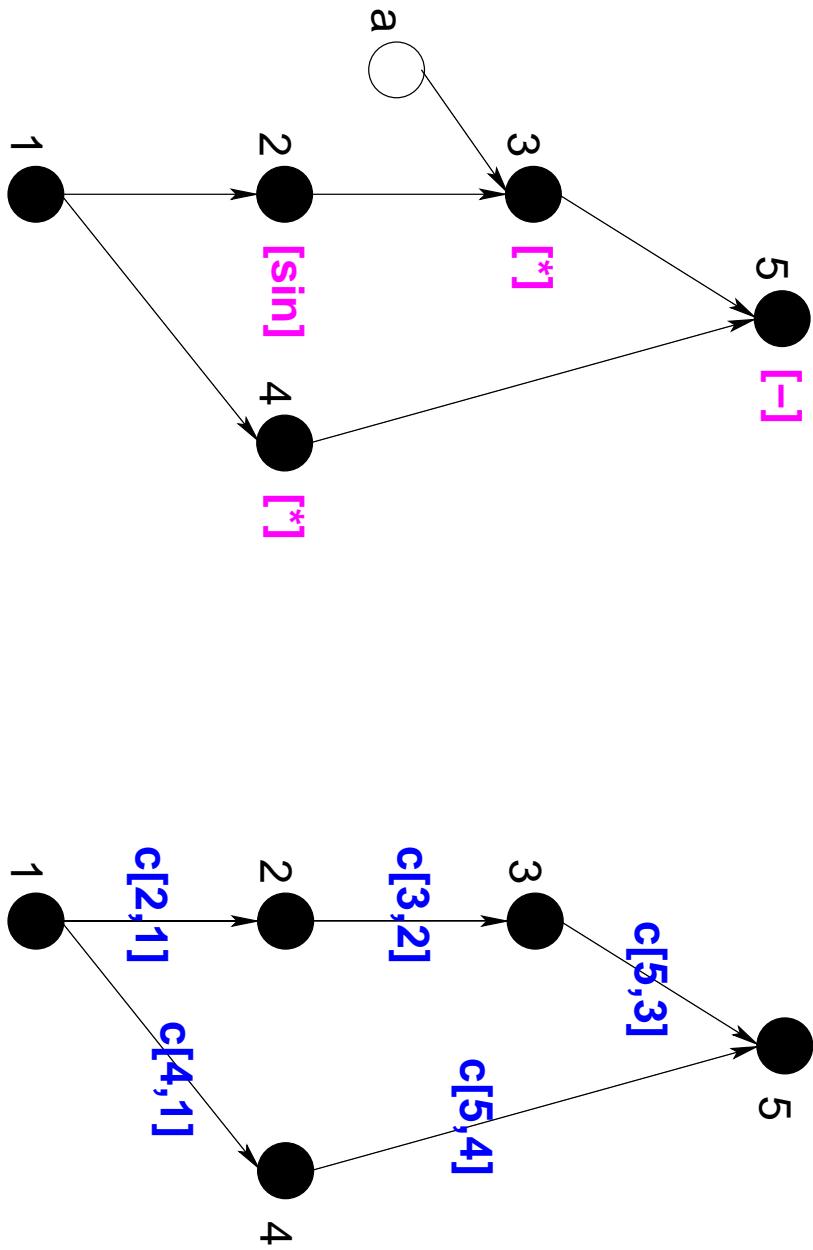
$$c_{j,i} = \frac{\partial \varphi_j}{\partial v_i}$$

for $j = 1, \dots, n + p + m$ and $i \prec j$. For example

$$\boxed{c[2,1]=\cos(x); \quad c[3,2]=a; \quad c[4,1]=2*x; \quad c[5,3]=1; \quad c[5,4]=-1}.$$

Linearized Computational Graph (of $y=a*\sin(x)-x*x$)

```
v1=x; v2=sin(v1); v3=a*v2; v4=v1*v1; v5=v3-v4; y=v5
```



Where are we?

- Introduction
- Forward and Reverse Modes
- Tool Development and Handout
- Advanced Issues
- Summary

Tangent-Linear Models – Forward Mode

Semantic program transformation

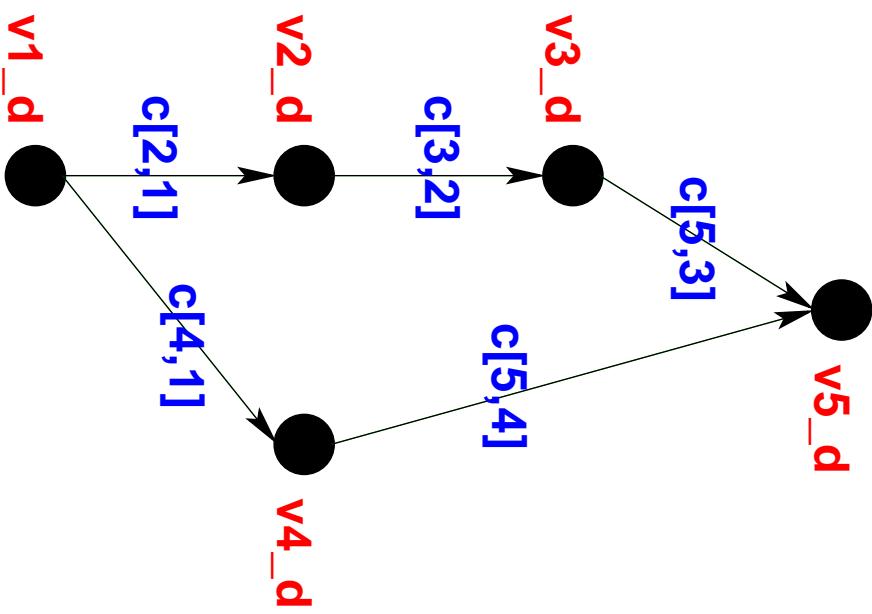
$$(y, \tilde{y}) = F(x, \tilde{x}) \rightarrow \dot{y} = \dot{F}(x, \tilde{x}, \dot{x})$$

building on local directional derivatives

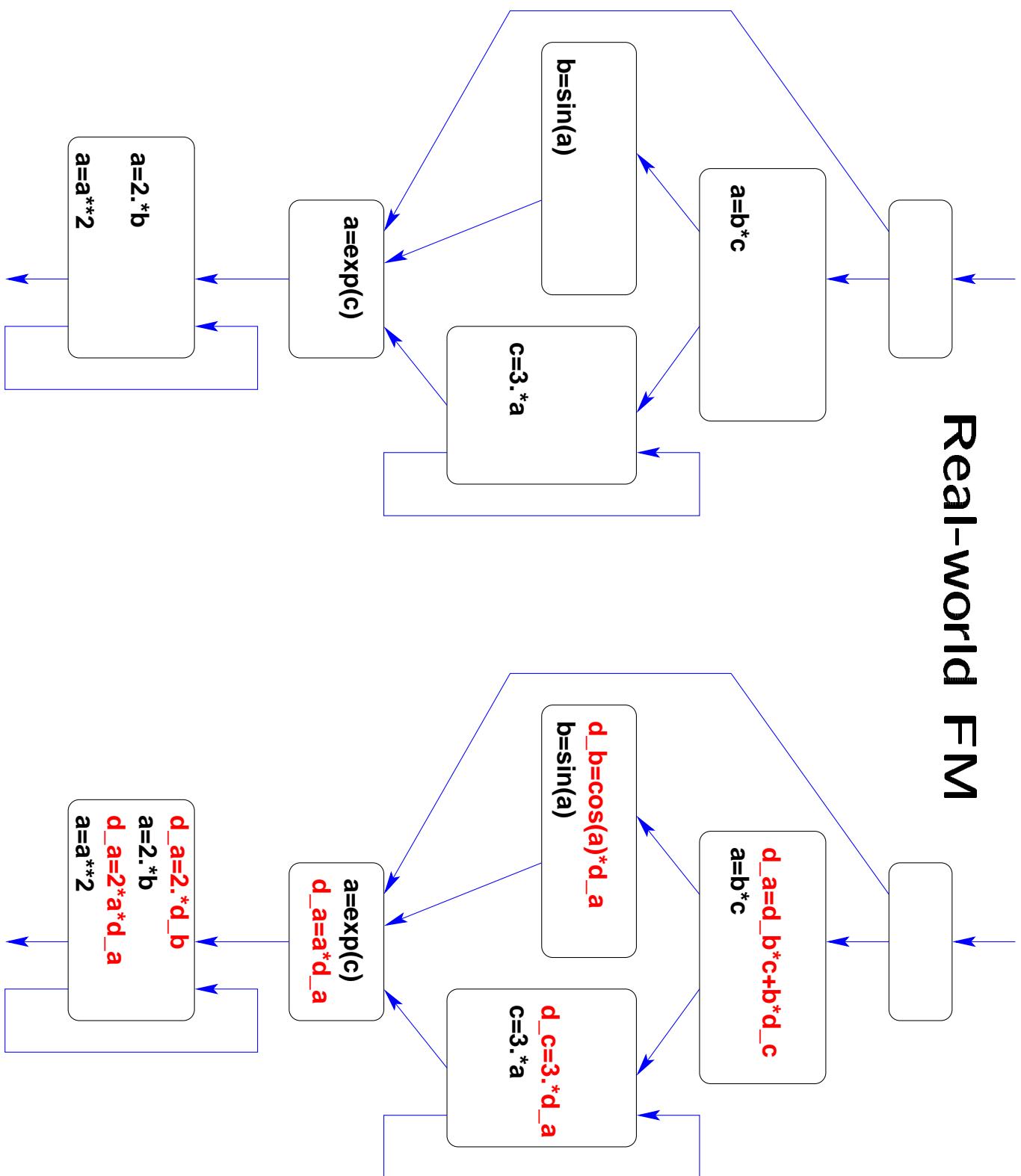
$$\begin{aligned}\dot{v}_j &= \sum_{i \prec j} c_{j,i} \cdot \dot{v}_i = (f'_j)^T \cdot (\dot{v}_i)_{i \prec j} , \\ \text{for } j &= 1, \dots, n + p + m .\end{aligned}$$

For example,

```
v1_d = x_d
v2_d = c[2,1] * v1_d
v3_d = c[3,2] * v2_d
v4_d = c[4,1] * v1_d
v5_d = c[5,3] * v3_d + c[5,4] * v4_d
v5_d = v5_d
```



Real-world FM



Adjoint Models – Reverse Mode

Semantic program transformation

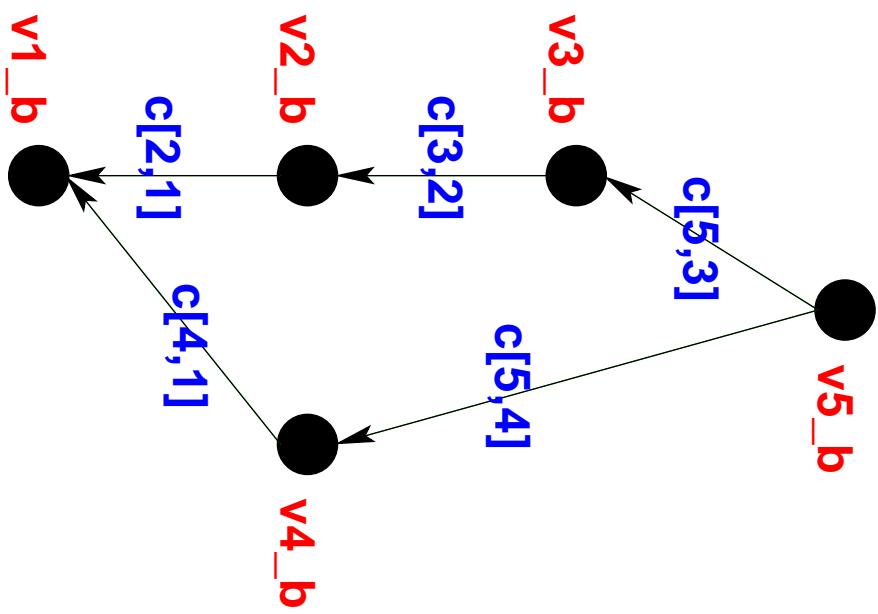
$$(y, \tilde{y}) = F(x, \tilde{x}) \rightarrow \bar{x} = \bar{F}(x, \tilde{x}, \bar{y})$$

building on local adjoints

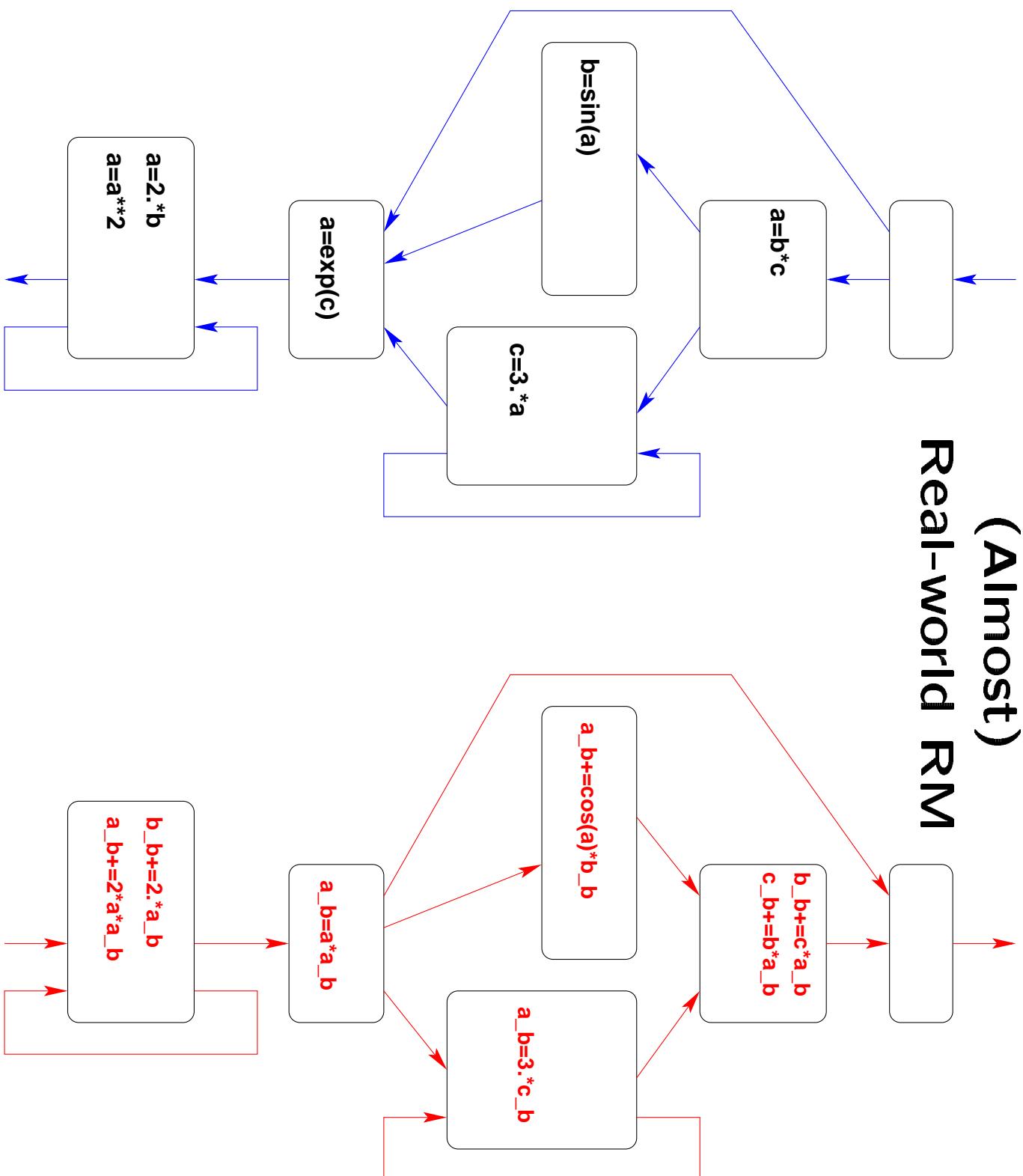
$$\begin{aligned}\bar{v}_j &= \sum_{k:j \prec k} c_{k,j} \cdot \bar{v}_k = \bar{f}_j \cdot (\bar{v}_k)_{\{k:j \prec k\}} , \\ \text{for } k &= n + p + m, \dots, 1 .\end{aligned}$$

For example,

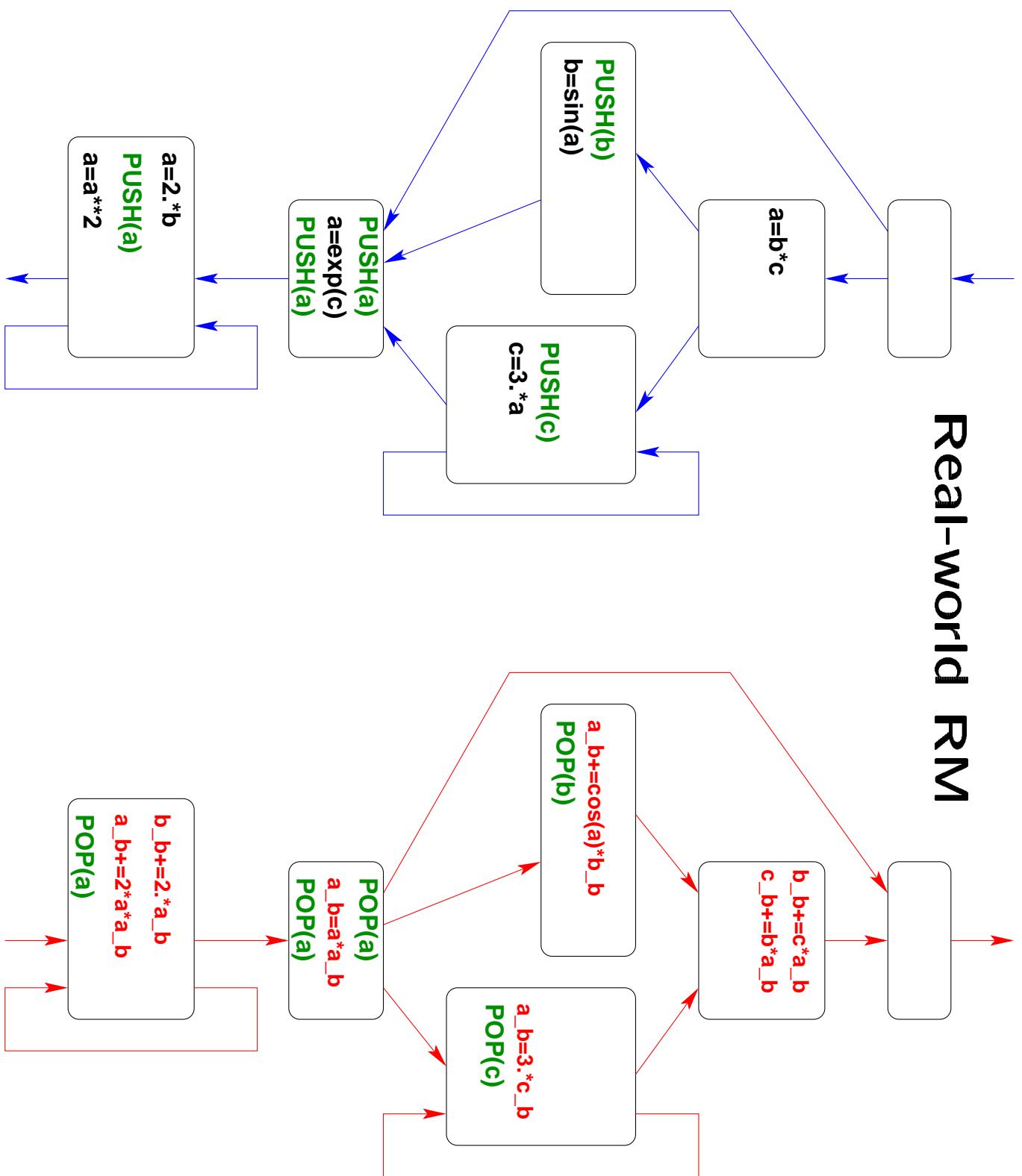
```
v5_b+=y_b
v4_b+=c[5,4]*v5_b; v3_b+=c[5,3]*v5_b
v1_b+=c[4,1]*v4_b
v2_b+=c[3,2]*v3_b
v1_b+=c[2,1]*v2_b
x_b=v1_b
```



(Almost) Real-world RM



Real-world RM



Where are we?

- Introduction
- Forward and Reverse Modes
- **Tool Development and Handout**
- Advanced Issues
- Summary

AD Tools (overloading)

Introduction of *active* floating point type (v, \dot{v}) and overloading of elemental functions

$$v = \varphi(u) \rightarrow (v, \dot{v}) = \varphi(u, \dot{u}) \quad .$$

For example,

$$v = u_1 \cdot u_2 \quad \rightarrow \quad (v, \dot{v}) = (u_1 \cdot u_2, \quad u_2 \cdot \dot{u}_1 + u_1 \cdot \dot{u}_2)$$

or

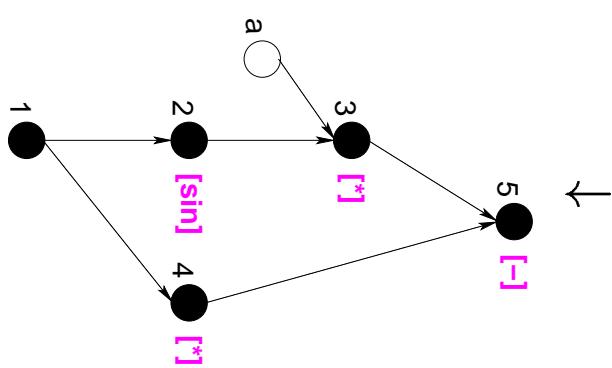
$$v = \cos(u) \quad \rightarrow \quad (v, \dot{v}) = (\cos(u), -\sin(u) \cdot \dot{u}) \quad .$$

AD Tools (Source Transformation)

$F \rightarrow$

$y = a * \sin(x) - x * x$

1. Lexical analysis
2. Syntax analysis
3. Semantic analysis
4. Static data flow analyses
5. AD
6. Code optimization
7. Unparsing



\downarrow

$$\rightarrow \begin{cases} \dot{F} \\ \bar{F} \\ etc. \end{cases}$$

$y_d = (a * \cos(x) - 2 * x) * x_d$

$y = a * \sin(x) - x * x$

Handout

- ADIC 1.1 (FM), ADIFOR 2.0 (FM), ADOL-C (FM,RM,
TAPENADE (FM,RM)
- Web servers available
- Examples under www.mcs-unix.anl.gov/~naumann/ad_tools.html
- AD literature
- Curtis-Powell-Reid Seeding [8]

Where are we?

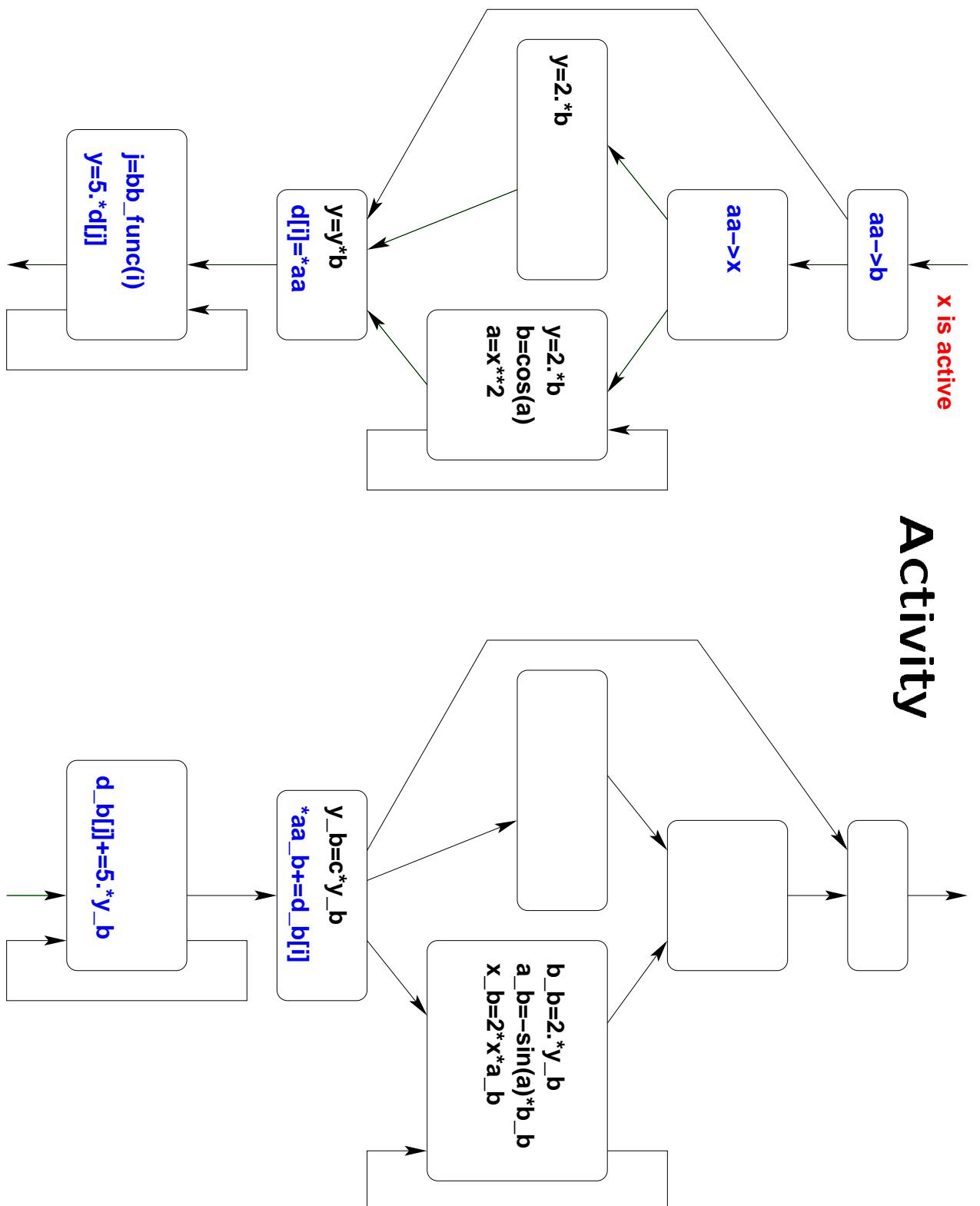
- Introduction
- Forward and Reverse Modes
- Tool Development and Handout
- Advanced Issues
- Summary

Advanced Issues

- Activity and TBR analyses
- Checkpointing
- Sparsity
- Preaccumulation
- Higher derivatives

x is active

Activity



Where are we?

- Introduction
- Forward and Reverse Modes
- Tool Development and Handout
- Advanced Issues
- Summary

Summary

- Tangent-linear and adjoint models via AD
- Cheap gradients in reverse mode
- Linearized single assignment form and computational graph
- Semantic transformation of numerical programs
- AD tools
- Advanced issues